BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Penelitian

Berikut ini merupakan hasil penelitian terdahulu, yang dijadikan sebagai kajian referensi bagi peneliti dalam melakukan penelitian selanjutnya. Refrensi tersebut diambil dari jurnal-jurnal yang berkaitan dengan peneltian mengenai sistem pencatatan dan monitoring produksi. Penelitian tersebut di antaranya adalah sebagai berikut :

2.1.1 Kajian Jurnal Pertama

Pada bulan Oktober 2022 *Mohamad Adriansyah* dan tim penyusun dalam jurnal ilmu komputer dan science dengan judul **Perancangan Sistem Informasi Pengelolaan Aset Inventaris Divisi Parkir Menggunakan *QR CODE* Berbasis *Web* (Studi Kasus: Universitas Pamulang)** membahas tentang *design* dan implementasi sistem informasi manajemen aset inventaris untuk divisi parkir Universitas Pamulang menggunakan kode QR berbasis *web* terkomputerisasi. Sistem ini dimaksudkan unutuk mempermudah dan mempercepat proses pengecekan kondisi dengan *QR Code*, mengetahui jumlah aset yang masuk atau keluar sehingga sistem yang dibangun diharapkan dapat memberikan informasi yang cepat dan akurat. Sistem ini dikembangkan menggunakan Bahasa pemograman *PHP* dan *database mysql* dengan metode pengembangan menggunakan model *waterfall* . Hasil jurnal menunjukan bahwa sistem terintegrasi dengan *database*, membuat proses pelaporan aset inventaris mudah dilakukan. Sistem mencatat semua barang masuk dan keluar beserta statusnya, sehingga memudahkan petugas untuk memulihkan data dalam mengetahui jumlah aset dalam bentuk URL yang terhubung langsung ke sistem dan di ubah dengan kode QR [2].

2.1.2 Kajian Jurnal Kedua

Pada bulan Desember 2020 *Siti Aminah* dan tim penyusun dalam jurnal Teknologi Rekayasa dengan judul **Penerapan Quick Response Code pada Digitalisasi Inventaris Laboratorium Berbasis *Android*** bertujuan untuk mengembangkan sistem digitalisai inventaris peralatan praktikum di laboratorium Teknik otomasi dan mekatronika di Politeknik Manufaktur Bandung dengan memanfaatkan teknologi industri 4.0. Laboratorium tersebut memiliki banyak peralatan praktikum, namun aktivitas inventarisasi yang dilakukan secara manual memakan waktu yang lama untuk mendapatkan hasil inventaris. Penelitian menggunakan metode *waterfall* untuk merancang sistem dan perangkat lunak digitalisasi inventaris. Hasil pengujian aplikasi digitalisasi inventaris laboratorium berbasis *QR Code* menunjukan bahwa proses inventarisasi memakan waktu sekitar satu menit. Dengan adanya sistem digitalisasi inventaris ini, dapat mempercepat aktivitas inventarisasi peralatan praktikum mahasiswa dan meningkatkan efisiensi pengelolaan inventaris [3].

2.1.3 Kajian Ketiga

Pada bulan Juli 2022 *M Irsyad Ramadhan* dan tim penyusun dalam jurnal *Journal Of Computer Science And Informatics Engineering* dengan judul **Pemanfaatan Barcode Generator Pada Aplikasi Manajemen Inventaris Barang Berbasis *Android* Di BPKH 1 Medan** bertujuan untuk menyajikan pengembangan aplikasi manajemen inventaris berbasis *Android* dengan kemampuan pembuatan *barcode QR* untuk mengatasi tantangan yang dihadapi oleh perusahaan dan institusi yang masih mengandalkan manual dan *book* atau sistem manajemen inventaris berbasis *Excel*. Jurnal ini menguraikan proses pengembangan menggunakan metode *Extreme Programming* (XP), yang meliputi perencanaan, desain, pengkodean, dan pengujian. Metode pengumpulan data yang digunakan dalam penelitian ini meliputi observasi, wawancara, dan tinjauan pustaka. Makalah ini menyimpulkan dengan menyatakan bahwa aplikasi yang dikembangkan dapat membantu kepala departemen peralatan dalam mengelola inventaris dan memantau tingkat stok melalui *smartphone Android* [4].

2.2 Landasan Teori

2.2.1 Sistem

Sistem adalah seperangkat komponen yang saling berhubungan dan saling bekerja sama untuk mencapai beberapa tujuan. Selain itu pengertian yang lain sistem terdiri dari unsur-unsur dan masukan (*input*), pengolahan (*processing*), serta keluaran (*output*). Dengan demikian,secara sederhana sistem dapat diartikan sebagai kumpulan atau himpunan dari unsur atau variabel-variabel yang terorganisasi, saling berinteraksi dan saling bergantung satu sama lain. Sistem di desain untuk memperbaiki atau meningkatkan pemprosesan informasi. Setelah dirancang, sistem diperkenalkan dan diterapkan ke dalam organisasi penggunanya. Jika sistem yang diterapkan itu digunakan maka implementasi sistem dapat dikatakan berhasil. Sedangkan jika para penggunanya menolak sistem yang diterapkan, maka sistem itu dapat digolongkan gagal [5].

2.2.2 Informasi

Informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam mengambil keputusan saat ini atau mendatang. Sedangkan Mc Leod (2001) mengatakan bahwa “Informasi adalah data yang telah diproses, atau data yang memiliki arti. Informasi juga merupakan salah satu sumber data yang tersedia bagi menejer dan dapat dikelola seperti halnya sumber daya yang lain”.

Berdasarkan berbagai definisi tersebut disimpulkan bahwa informasi adalah data yang diolah dan berguna bagi pemakainya dalam pengambilan keputusan. Informasi yang baik adalah informasi yang memberikan nilai tambah *(value added)* bagi pemakainya. Pemakai akan menggunakan informasi untuk perencanaan, koordinasi, evaluasi dan pengambilan keputusan. Oleh karena itu informasi harus mempunyai ciri-ciri, yaitu dapat mengurangi ketidakpastian,dapat menggambarkan adanya berbagai peluang dan dapat mengevaluasi hasil [6].

2.2.3 *QR Code*

QR Code adalah gambar berformat matriks dua dimensi yang memiliki kapabilitas untuk menyimpan data di dalamnya. *QR Code* merupakan perkembangan dari kode batang (*barcode*). *Barcode* adalah simbol yang digunakan untuk mengidentifikasi objek fisik yang terdiri dari pola batang hitam dan putih untuk memudahkan pengenalan oleh perangkat komputer.

Gambar 2. 1 *QR Code*

QR Code adalah singkatan dari *Quick Response Code*, atau dalam bahasa Indonesia dapat diartikan sebagai "Kode Respon Cepat." *QR Code* dikembangkan oleh *Denso Corporation*, sebuah perusahaan Jepang yang berfokus pada industri otomotif. *QR Code* ini pertama kali diperkenalkan pada tahun 1994 dengan tujuan awalnya digunakan untuk pelacakan kendaraan dalam proses manufaktur secara efisien serta mendapatkan *respons* dengan cepat [6].

2.2.4 Inventarisasi

Istilah "inventarisasi" berasal dari kata "inventaris," yang merujuk pada daftar atau pencatatan barang-barang. Inventarisasi adalah proses yang melibatkan pencatatan dan penyusunan barang-barang atau bahan yang ada dengan benar sesuai dengan ketentuan yang berlaku. Inventarisasi memiliki tujuan utama dalam meningkatkan pengurusan dan pengawasan yang efektif terhadap barang-barang, baik milik negara maupun swasta. Proses inventarisasi juga memberikan wawasan berharga yang dapat meningkatkan efisiensi pengelolaan sarana dan prasarana [7].

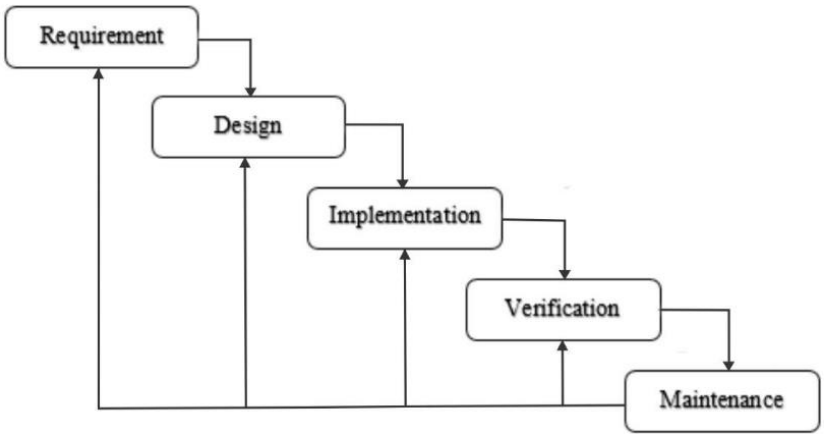
2.2.5 Metode Pengembangan Sistem

Sistem yang sedang berjalan atau sedang digunakan oleh organsisi atau perusahaan akan terus dikembangkan untuk memperbaiki kekurangan-kekurangan pada sistem tersebut. Untuk melakukan pengembangan sistem, metode yang digunakan adalah SDLC. Metode adalah tahap-tahap ataupun aturan untuk melakukan sesuatu. McLeod dan Schell (2007) mengatakan metode adalah cara untuk melakukan sesuatu. SDLC adalah sebuah metode yang digunakan untuk mengembangkan sebuah sistem. SDLC adalah sebuah proses logika yang digunakan oleh seorang sistem analist untuk mengembangkan sebuah sistem informasi yang melibatkan *requirements, validation, training* dan pemilik sistem [8].

System Development Life Cycle (SDLC) atau siklus hidup pengembangan sistem dalam rekayasa sistem dan rekayasa perangkat lunak adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. SDLC juga merupakan pola untu mengembangkan sistem perangkat lunak yang dterdiri dari tahapan perencanaan (*planning*), analisis (*analyst*), desain (*design*), implementasi (*implementation*), uji coba (*testing*) dan pengelolaan (*maintenance*) [9].

2.2.6 Metode *Waterfall*

Pengertian Metode *Waterfall* atau metode air terjun atau yang sering disebut metode *waterfall* sering dinamakan siklus hidup klasik (*classic life cycle*), dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (*planning*), permodelan (*modeling*), konstruksi (*construction*), serta penyerahan sistem ke para pelanggan/pengguna (*deployment*), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Pressman, 2012). Tahapan metode *waterfall* dapat dilihat pada gambar di bawah ini. Berikut tahapan - tahapan air terjun :



Gambar 2. 2 Metode *waterfall*

1. *Requirement Analysis*

   Pada tahap ini, sebagai pembuat aplikasi diharuskan memahami perangkat lunak yang diharapkan oleh pengguna dan batasan aplikasi tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

2. *System Design*

   Selanjutnya pada tahap ini, spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan desain aplikasi disiapkan. Desain Sistem membantu dalam menentukan perangkat keras (*hardware*) dan sistem persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan serta pembuatan desain rancangan aplikasi yang akan dibuat.

3. *Implementation*

   Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut unit, yang terintegrasi dalam tahap selanjutnya. Setiap unit dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai unit *testing.*

4. *Integration & Testing*

   Seluruh unit yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing unit.

Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

5. *Operation & Maintenance*

   Tahap akhir dalam model *waterfall* . Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru [10].

2.2.7 UML *(Unified Modeling Language)*

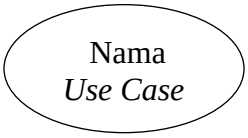
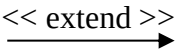
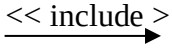
Unified Modeling Language (UML) adalah sebuah bahasa yang berdasarkan gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis Objek. *Unified Modeling Language* (UML) bukanlah merupakan bahasa pemprograman tetapi model-model yang tercipta berhubungan langsung dengan berbagai macam bahasa pemprograman, sehingga memungkinkan melakukan pemetaan (mapping) langsung dari model-model yang dibuat dengan *Unified Modeling Language* (UML) dengan bahasa-bahasa pemprograman berorientasi obyek [11].

UML *(Unified Modeling Language)* adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisa & desain, serta menggambarkan arsitektur dalam pemrograman berorintasi objek. UML *(Unified Modeling Language)* memiliki diagram-diagram yang digunakan dalam pembuatan aplikasi berorientasi objek [12], diantaranya :

a. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk sistem informai yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.Berikut adalah simbol-simbol yang ada pada diagram *use case* [12];

Tabel 2. 1 Tabel Simbol - Simbol *Use Case Diagram*

Simbol	Deskripsi
Use case 	Fungsionalisasi yang di sediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya di nyatakan dengan menggunakan kata kerja di awal *frase name use case*.
Aktor /*actor*  Nama Aktor	Orang, proses atau sistem yang lain yang berinteraksi dengan sistem informasi yang akan di buat diluar sistemyang akan di buat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tetapi aktor belum tentu menggunakan orang, biasanya di nyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / *association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang di tambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu; mirip dengan prinsip *inheritance* pada pemograman berorientasi objek, biasanya use case tambahan memiliki nama depan yang sama dengan use case yang di tambahakan misal arah panah mengarah pada use case yang di tambahakan
Generalisasi 	Hubungan generalisasi dan spesialisasi antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Include 	Relasi *use case* tambahakan ke subuah *use case* dimana *use case* di tambhakan memerlukan *use*

	case ini menjalakan fungsinya atau syarat di jalankan *use case* ini

b. Actvity Diagram

Diagram aktivitas menggambarkan aliran kerja (*workflow*) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah diagram aktivitas menggambarkan aktivitas sistem, bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas[12];

Tabel 2. 2 Tabel Simbol-Simbol Activity Diagram

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memilih sebuah status awal.
Aktivitas 	Aktivitas yang di lakukan sistem aktivitas biasanya di awali dengan kata kerja
Percabangan / decision 	Asosiasi percabangan dimana ada pilihan aktivitas lebih dari
Penggabungan / join 	Asosiasi penggabungan dimana lebih satu aktivitas di gabung menjadi satu
Status akhir 	Status akhir yang di lakukan sistem sebuah diagram aktivitas memiliki sebuah status akhir

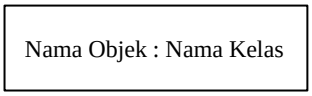
c. *Sequence* Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Maka dari itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* serta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu [12].

Sequence diagram mendokumentasikan komunikasi antar kelas-kelas. Diagram ini menunjukkan sejumlah obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek didalam *use case* [13].

Dalam diagram ini, kelas dan aktor ditempatkan secara berurutan dari kiri ke kanan di bagian atas diagram, dengan garis diposisikan secara vertikal di depan kelas dan aktor. Berikut adalah simbol-simbol yang ada pada *Sequence* diagram;

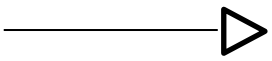
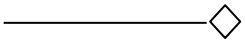
Tabel 2. 3 Tabel Simbol-Simbol *Sequence* Diagram

Simbol	Deskripsi
Aktor  Nama Aktor	Orang, proses atau sistem yang lain yang berinteraksi dengan sistem informasi yang akan di buat diluar sistemyang akan di buat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tetapi aktor belum tentu menggunakan orang; biasanya di nyatakan menggunakan kata bemda di awal frase nama aktor
Garis hidup / *lifeline*	Menyatakan hidup suatu objek
Objek 	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe *create* 	Objek yang lain, arah panah mengarah pada objek yang di buat

d. Class Diagram

Diagram kelas menunjukan interaksi antara kelas dalam sistem. Kelas mengandung informasi dan tingkah laku yang berkaitan dengan informasi tersebut. Sebuah kelas pada diagram kelas dibuat untuk setiap tipe objek pada diagram sekuensial atau diagram kolaborasi [12]. *Class diagram* menggambarkan struktur sistem dari segi pendefinisian yang akan dibuat untuk membangun sistem. Berikut adalah simbol-simbol yang ada pada diagram kelas.

Tabel 2. 4 Tabel Simbol-Simbol Class Diagram

Simbol	Deskripsi
Kelas	Kelas pada struktur sistem
Antarmuka/ / *interface*  Nama_*interface*	Sama dengan konsep interface dalam pemograman berorientasi objek
Asosiasi / *association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga di sertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesifikasi (umum-khusus)
Keberuntung/*dependency* 	Relasi antar kelas dengan makna ketergantungan antar kelas
Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua bagian

2.2.8 *Android*

Android adalah sistem operasi *open source* yang berbasis linux dengan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi baru [14]. *Android* adalah sistem operasi untuk telepon seluler yang berbasis Linux. *Android* menyediakan *platform* terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Dari kesimpulan diatas penulis menyimpulkan *Android* merupakan sistem yang beroprasi untuk perangkat *smartphone* yang berbasis linux [15]. Berikut gambar *Android* :

Gambar 2. 3 *Android*

2.2.9 *Android* Studio

Android Studio adalah IDE (*Integrated Development Environment*) resmi yang didukung Google untuk mengembangkan aplikasi *Android* berbasis Intellij IDEA. Sebagai editor kode dan alat bantu pengembang yang *powerful*, *Android Studio* memberikan fitur-fitur yang akan meningkatkan produktivitas ketika membangun aplikasi *Android*, seperti:

- Sistem gradle yang fleksibel
- Emulator dengan berbagai macam fitur
- Kerangka kerja dan peralatan *testing* yang luas
- Peralatan lint untuk mendapatkan performa, penggunaan, dukungan versi dan masalah-masalah lain
- Dukungan C++ dan NDK
- Didukung oleh *Google Cloud Platform*, memudahkan dalam integrasi *Google Cloud Messaging* dan *App Engine* [16].

2.2.10 *Dart*

Dart merupakan Bahasa pemrograman umum yang dibuat oleh Google pada tahun 2011, bahasa ini digunakan dalam membuat aplikasi *Flutter*. Mirip seperti *java* dan C#, *Dart* memiliki sintaks yang mirip dengan C. Tidak seperti Bahasa pemrograman lainnya, Bahasa ini dioptimasi untuk dikembangkan dan dijalankan di berbagai *platform :*

- Di *Web browser* sebagai *JavaScript*

- Sebagai Aplikasi *Native*

Dart dirilis secara *open source* sehingga dapat digunakan secara bebas , berbasis kelas dan *orientasi object* serta menggunakan sintaks Bahasa perograman C [16].

2.2.11 *Flutter*

Flutter adalah sebuah teknologi yang dibuat oleh Google untuk membangun aplikasi *Mobile* (*Android*, iOS), *web* dan *desktop* (*Windows, Mac, Linux*) sekaligus dengan hanya melalui satu kode sumber saja, kode ini akan *di-compile* secara *native*. Poin ini yang menjadi perbedaan antara Google *Flutter* dan *Framework cross- platform* lainnya. *Flutter* menggunakan bahasa pemrograman *Dart* [16].

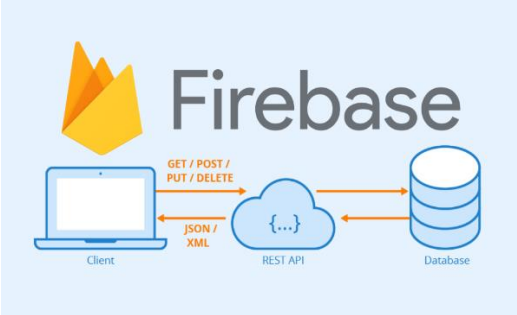
Penulis menyimpulkan *Flutter* merupakan *Framework* atau kerangka kerja untuk membuat aplikasi *Mobile*, *web* dan *desktop* dengan satu *codebase* yaitu dengan bahasa pemrograman *Dart* .

2.2.12 *Firebase*

Firebase adalah penyedia layanan *cloud* dengan *back-end* sebagai servis yang berbasis di San Fransisco, California. *Firebase* membuat sejumlah produk untuk pengembangan aplikasi *Mobile* ataupun *web*. *Firebase* didirikan oleh Andrew Lee dan James Tamplin pada tahun 2011 dan diluncurkan dengan *cloud database* secara *realtime* di tahun 2012. Produk utama dari *Firebase* yakni suatu *database* yang menyediakan API untuk memungkinkan pengembang menyimpan dan mensinkronisasi data lewat *multiple client*. Perusahaan ini diakusisi oleh Google pada Oktober 2014. *Firebase* adalah penyedia layanan *realtime database* dan backend sebagai layanan. Suatu aplikasi yang memungkinkan pengembang membuat API untuk disinkronisasikan untuk client yang berbeda-beda dan disimpan pada cloud. *Firebase* memiliki banyak library yang memungkinkan untuk mengintegrasikan dengan *Android, Ios, Javacript, Java, Objective-C dan Node.JS* . *Database Firebase* juga bisa diakses lewat *REST API. REST API* tersebut menggunakan protokol *Server-Sent Event* dengan membuat koneksi HTTP untuk menerima push notification dari server. Pengembang menggunakan *REST API*

untuk post data yang selanjutnya *Firebase client library* yang sudah diterapkan pada aplikasi yang dibangun yang akan mengambil data secara *realtime* [17].

Dari pengertian diatas penulis menyimpulkan *Firebase* adalah penyedia layanan *cloud* yang memungkinkan pengembang untuk membangun aplikasi *Mobile* dan *web* dengan menyediakan berbagai produk, termasuk *database real-time*. Produk ini memungkinkan sinkronisasi data melalui berbagai klien dan tersedia dengan berbagai *library* untuk berbagai *platform*. *Firebase* juga dapat diakses melalui *REST API* dan *Server-Sent Event* untuk pembaruan data *real-time*. Berikut gambaran dari topologi *Firebase* :



Gambar 2. 4 Topologi *Firebase*

2.2.13 Visual Studio Code

Visual studio code yang lebih dikenal dengan *vs code* adalah sebuah editor kode gratis, lintas *platform* yang berjalan secara *native* di sistem operasi OS X, Linux dan Windows. Visual studio code editor yang dikembangkan oleh Microsoft ini memiliki karakteristik *independen*, ringan dan tidak bercampur dengan Visual Studio IDE [16].

2.2.14 Blackbox Testing

Metode *Blackbox Testing* adalah sebuah metode yang dipakai untuk menguji sebuah software tanpa harus memperhatikan *Detail software*. Pengujian ini hanya memeriksa nilai keluaran berdasarkan nilai masukan masing- masing. Tidak ada upaya untuk mengetahui kode program apa yang output pakai (Latif, 2015). Proses *Black Box Testing* dengan cara mencoba program yang telah dibuat dengan mencoba memasukkan data pada setiap formnya. Pengujian ini diperlukan untuk mengetahui program tersebut berjalan sesuai dengan yang dibutuhkan oleh perusahaan [18].